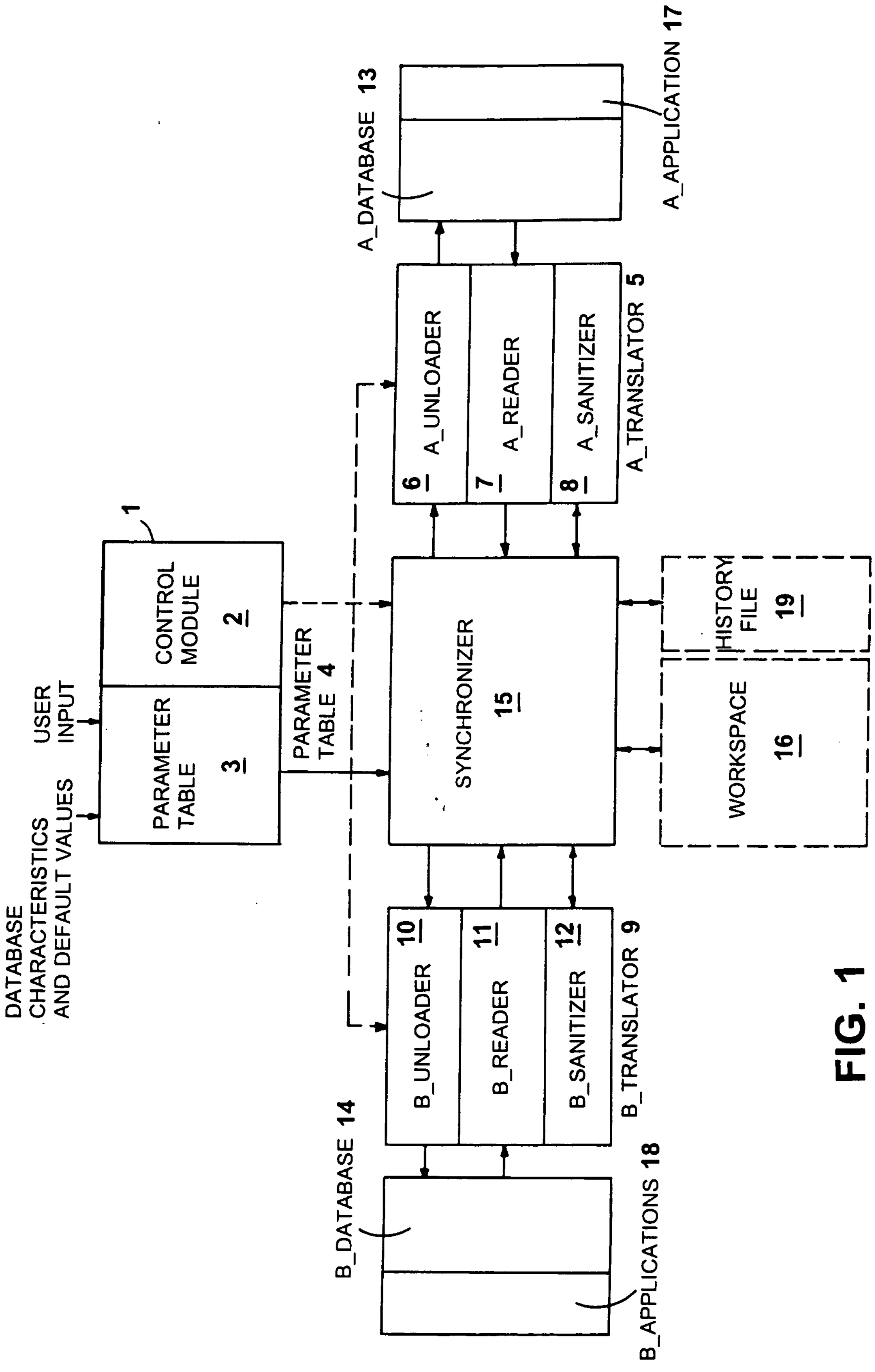

DATABASE CHARACTERISTICS AND DEFAULT VALUES
USER INPUT
1
PARAMETER TABLE 3
CONTROL MODULE 2
PARAMETER TABLE 4
SYNCHRONIZER 15
HISTORY FILE 19
WORKSPACE 16
6 A_UNLOADER
7 A_READER
8 A_SANITIZER
A_TRANSLATOR 5
A_DATABASE 13
A_APPLICATION 17
10 B_UNLOADER
11 B_READER
12 B_SANITIZER
B_TRANSLATOR 9
B_DATABASE 14
B_APPLICATIONS 18

FIG. 1

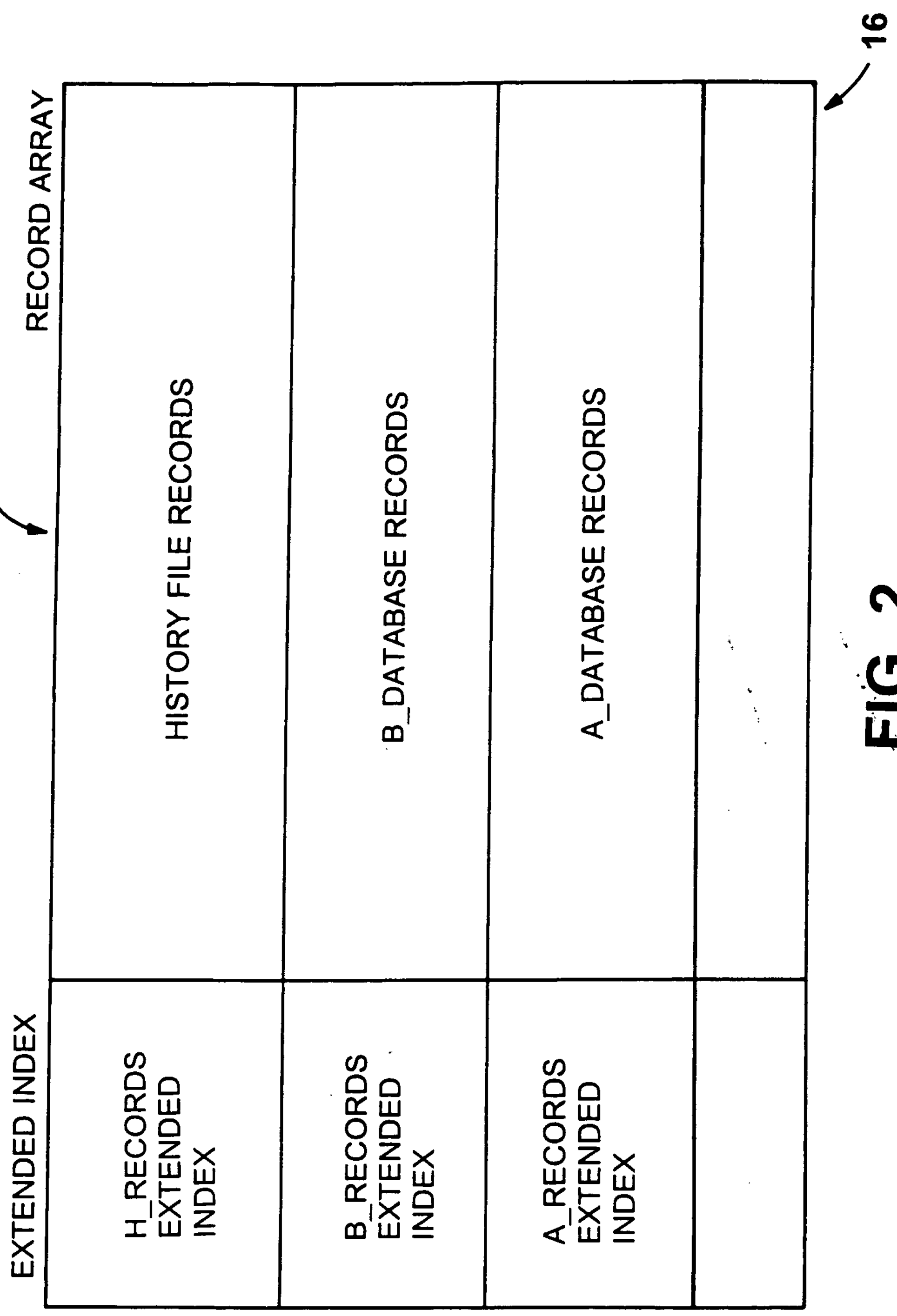

RECORD ARRAY
EXTENDED INDEX
21
16
HISTORY FILE RECORDS
B_DATABASE RECORDS
A_DATABASE RECORDS
H_RECORDS EXTENDED INDEX
B_RECORDS EXTENDED INDEX
A_RECORDS EXTENDED INDEX

FIG. 2

Pseudo Code for Translation Engine Control Module

```
100.    CREATE Parameter_Table from User Input A & B database characteristics and default values
101.    INSTRUCT Synchronizer to initialize itself
102.    INSTRUCT Synchronizer to LOAD the History_File into its WORKSPACE
103.    INSTRUCT B_Translator to LOAD all of B_records from B_Database and SEND to Synchronizer
        (Synchronizer STORES these records in WORKSPACE)
104.    INSTRUCT A_Translator to SANITIZE B_records that were just LOADED (A_Translator USES
        Synchronizer services to read and write records in the WORKSPACE; Synchronizer maps these records
        using the B-A_Map before sending them to A_Translator and maps them back using A-B_Map before
        rewriting them into the WORKSPACE)
105.    INSTRUCT A_Translator to LOAD all of A_records from A_Database and SEND to Synchronizer
        (Synchronizes STORES these records in WORKSPACE by first mapping then using the A-B_Map and
        them storing in their new form)
106.    INSTRUCT B_Translator to SANITIZE A_records that were just LOADED (B_Translator uses
        Synchronizer services to read and write records in the WORKSPACE)
107.    INSTRUCT Synchronizer to do CAAR (Conflict Analysis And Resolution) on all the records in
        WORKSPACE.
108.    INFORM user exactly what steps Synchronizer proposes to take (i.e. Adding, Changing, and Deleting
        records). WAIT for User
109.            IF user inputs NO, THEN ABORT
110.    INSTRUCT B_Translator to UNLOAD all applicable records to B_Database.
111.    INSTRUCT A_Translator to UNLOAD all applicable records to the A_Database.
112.    INSTRUCT Synchronizer to CREATE a new History File.
```

FIG. 3

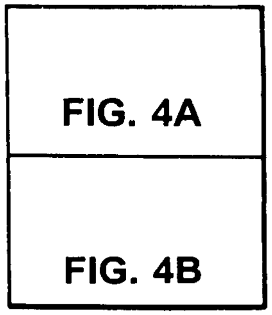


FIG. 4

Pseudocode for Generating Parameter Table

```
{Get Input from the user}
150.    ASK user to whether to synchronize based on a previously stored set of preferences
        (Previous_Preferences) or based on a set of new preferences (New_Preferences)
151.    IF New_Preferences THEN
152.            ASK user whether Incremental_Synchronization or Synchronization_from_Scratch
153.            ASK user following information and STORE in Parameter_Table
        a.      A_Application and B_Application Names
        b.      ADB and BDB Names
        c.      ADB and BDB Locations
        d.      Which sections to Synchronize
        e.      Conflict Resolution Option: IGNORE, ADD, DB WINS, BDB WINS, or NOTIFY
        f.      Other user preferences
154.            ASK user whether wants default mapping for the selected sections of the two databases or wants
                to modify default mapping
155.            LOAD A_Database→B_Database (2)
156.            IF Default_Mapping THEN
157.                    STORE A→B_Map AND B→A_Map in Parameter_Table
158.            END IF
159.            IF Modified_Mapping THEN
160.                    DISPLAY A→B_Map and B→A_Map
161.                    ASK user to modify Maps as desired
162.                    STORE the new A→B_Map and B→A_Map in the Parameter_Table
163.            END IF
165.            END IF
```

FIG. 4A

```
166.  IF Previous_Preferences THEN
167.          ASK user whether Incremental_Synchronization or Synchronization_from_Scratch
168.          STORE in Parameter_Table
169.          LOAD Previous Preferences regarding which databases, mapping, and so on
170.          STORE in the Parameter_Table
171.  END IF
      {User now specifies Date Range}
172.  ASK user to choose Date Range Option
      a.      Previously chosen  Automatic_Date_ Range calculated from today
      b.      Input New Automatic_Date_Range
      c.      Input static Date Range for this Synchronization
      d.      All dates
173.  CALCULATE Start_Current_Date_Range and End_Current_Date_Range based on values from step 171
174.  STORE in Parameter_Table
175.  LOAD parameters setting out characteristics of A_Database and B_Database from Parameters database,
      including
      a.      Field_List_A and Field_List_B
      b.      A_Translator and B_Translator Module Identifiers
      c.      ADB_Section_Names and BDB_Section_Name
176.  STORE in Parameters Table
```

FIG. 4B

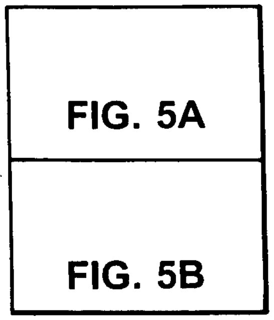


FIG. 5

```
200.    RECEIVE following from Parameter Table
        1) Name of A_App
        2) Name of B_App
        3) Name and Location  of A_DB
        4) Name and Location of B_DB
        5) Section name of A_Application to be synchronized
        6) Section name of B_Application to be synchronized
        7) Incremental_Synchronization or Synchronization_From_Scratch Flags
201.    SEARCH for H_File matching Parameters 1-6
202.    If Found H-File and Incremental_Synchronization THEN DO nothing
203.    IF Found H-File and Synchronization_from_Scratch, THEN DELETE H_File
204.    IF NOT found H-File, THEN SET Synchronization_from_Scratch AND ASSIGN file name for history
        file.
205.    LOAD from Parameter_Table Start_Current_Date_Range and End_Current_Date_Range
206.    LOAD from Parameter_Table Field_Lists for A-DB and B-DB and field and mapping information
207.    If Incremental_Synchronization THEN COMPARE Field_Lists and Maps from Parameter_Table with
        History_Field_Lists and Maps
208.    IF exact match THEN DO nothing
209.    IF not exact match THEN DELETE H_file AND SET Synchronization_from_Scratch
210.    CREATE WORKSPACE using Field_List_B
211.    If Incremental_Synchronization THEN Copy H_file into WORKSPACE
212.    FOR each H-Record update
{analyze & update source of extended index}
213.            Do Nothing to NEXT_IN_FIG
214.
```

FIG. 5A

```
215.            FIND H-Record with matching KeyFields
216.            IF FOUND THEN Update NEXT_IN_SKG of H-Record
217.            IF Appointment type and Non-Recurring record THEN
218.                IF (Start_Date after End_Previous_Date_Range) OR (End_Date before
                    Start_Previous_Date_Range) THEN SET Bystander Flag END IF
219.                IF (Start_Date after End_Current_Date_Range) OR (End_Date before Start_Current
                    _Date_Range) THEN SET Outside_Current_Range END IF
{Recurring records}
220.                ELSE
221.                Fan_Out_Recurrence_Pattern for H-Record
222.                SET Bystander Flag and Outside_Current_Range Flags for H-Record
223.                For all Fanned out Instances
224.                    IF (Start_Date Before End_Previous_Date_Range) OR (End_Date after
                        Start_Previous_Date_Range) THEN UN-SET Bystander Flag of Recurring H-
                        Record END IF
225.                    IF (Start_Date before End_Current_Date_Range) OR (End_Date after
                        Start_Current _Date_Range) THEN UN-SET Outside_Current_Range END IF
226.                END LOOP
227.            END IF
228.        END LOOP
```

FIG. 5B

```
235.    LOAD Rep_Basic, Start_Date, Stop_Date, Frequency
236.    CALCULATE Useful Start_Date and Useful_Stop_Date based on Start_Date, Stop_Date, Max_Fan_Out
        and Usefulness_Range_Future & Past
237.    REPEAT
238.            CALCULATE Next_Date based on Useful_Start_Date, Current_Date, Rep_Basic, Frequency,
                Max_Fan_Out
239.            IF Next_Date After Useful_Stop_Date, THEN EXIT
240.            STORE Next_Date
241.                    Fan_Out_Date_Array
242.            Current_Date = Next_Date
243.    END LOOP
```

FIG. 6

Pseudocode for Key_Field_Match

```
250.    RECEIVE Key_Field_Hash and WORKSPACE_ID
251.    For all records in WORKSPACE
252.            IF Match_Hash_Value equals Hash Values of Record THEN LOAD the two records
253.                    COMPARE the key fields two records
254.                    IF Exact Match THEN SET Match_Found
255.                    EXIT LOOP
256.            END IF
257.    END LOOP
258.    If Match_Found THEN SEND Success Flag and WORKSPACE ID of Matching record
```

FIG. 7

Pseudo Code for Loading Records of B_database into WORKSPACE

```
B_Translator:

300.    FOR ALL Records in B_DB
301.            READ Record from B_DB
302.            IF (record outside of combination of Current_Date_Range and Previous_Date_Range), THEN
                GOTO END LOOP
303.            IF NOT right origin tag for this synchronization THEN GOTO END LOOP
304.            SEND Record to Synchronizer 325-236
305.    END LOOP

Synchronizer:
325.    RECEIVE B_Record
326.    STORE in WORKSPACE in next available space
```

FIG. 8

Pseudo Code for Generic A_Sanitization of B_DB Records in Workspace

A_Translator:

```
350.  REPEAT
351.          FOR EVERY Field in an A_Record
352.                  REQUEST Field from Synchronizer
353.                  IF Last_Field, THEN EXIT LOOP
354.                  SANITIZE Field, according to A_Sanitization rules
355.          END LOOP
356.          IF Last_Field, THEN EXIT LOOP
357.          SANITIZE Record according to A_Sanitization rule
358.          FOR EVERY Field in an A_Record
359.                  SEND Field value to Sanitizer
360.          END FOR
361.  UNTIL EXIT
```

SYNCHRONIZER:

```
375.  In Response to Request for Field by A_Sanitizer
376.  REPEAT UNTIL LAST RECORD
377.          READ B_Record
378.          MAP Record according to B_A Map
379.          REPEAT UNTIL A_Translator Request a field from a new Record
380.                  SEND REQUESTED B_field to A_Translator
381.                  WAIT FOR RETURN of B_Field from A_Translator
382.                  STORE field Value in Mapping_Cache
383.          END LOOP
384.          MAP record in Cache according to A-B Map
385.          STORE record in WORKSPACE
386.  END LOOP
387.  SEND Last_Field flag in response to REQUEST
```

FIG. 9

```
Specific Example of Sanitization
400.    IF StartDate and EndDate are both blank
401.            Make Alarm Date blank and make Alarm Flag = FALSE
402.      ELSE IF EndDate is blank THEN SET EndDate = StartDate
403.            ELSE IF StartDate is blank OR is greater than EndDate THEN      SET StartDate =
                EndDate END IF

404.            IF AlarmFlag is TRUE and AlarmDate is blank THEN SET  AlarmDate = StartDate
405.                    ELSE IF AlarmDate is greater than EndDate THEN SET AlarmDate = EndDate
406.    END IF
```

FIG. 10

Pseudo_code for Orientation Analysis (Index Value analysis)

```
450.  FOR EVERY Record of database in WORKSPACE
451.      CALCULATE Key_Field_Hash from Section Subtype value for the record & all Mapped Key
          Fields
452.      CALCULATE Non_key_Fields_Hash from all Mapped Non_key Fields which are not marked as
          No_Reconcile
453.      CALCULATE Exclusion_List_Hash, if Recurring_Master, from Exclusion_List
454.      CALCULATE Non_Date_Hash from all non-date mapped non-key fields which are not
          No_Reconcile fields
455.      If B_Record THEN CALCULATE B_ID_Hash
456.      IF A_Record THEN CALCULATE A_DB ID Hash
457.      CALCULATE Start_Date_Time values (for Appointments and TO DO Lists)
458.      CALCULATE End_Date_Time
459.      IF Recurring Item and No instances in Current Date Range THEN SET Out_Of_Range
460.      IF (Start_Date After End_Current_Date_Range OR End_DateBefore Start_Current_Date_Range,
          THEN SET Out_Of_Range_Flag ELSE SET IN_Range Flag
461.      END IF
462.      IF Matching Unique ID in H_records THEN ADD to CIG
463.      IF Matching Unique_ID in H_records, THEN SET WARNING FLAG
464.      IF an H or current database record with same key field values (using Key_Field_Match function,
          Fig. 7), THEN ADD Current Record to SKG of the H or A_record
465.  END LOOP
```

FIG. 11

Pseudocode for Conflict Analysis And Resolution (CAAR)

```
500.    Analyze ID_Bearing FIGS.
501.    Analyze and expand ID_bearing CIGs
502.    Finding Matches between Recurring Items and Non-Unique ID bearing Instances
503.    Analyze SKGs
504.    SET CIG Types
```

FIG. 12

Pseudocode for Analyzing ID_bearing FIGs

```
550.  FOR EVERY Recurring Master of ID_Bearing FIGS in H_file
551.      FOR EVERY FIG H_Record in Recurring Master FIG
552.          REMOVE Record from SKG it belongs to
553.          IF Record is a singleton CIG, THEN ADD to New_Exclusion_List
554.          IF Record is a doubleton CIG, THEN
555.              IF the two Records in CIG are Identical, THEN remove other RECORD from
                  its SKG
556.              END IF
557.              ELSE IF the two records are NOT Identical, THEN ADD FIG record to
                  New_Exclusion_List and change records into singleton CIGs
558.          END IF
559.      END LOOP
560.      CREATE Synthetic Master record entry in WORKSPACE
561.      COPY value from one of the CIG mates into Synthetic Master
562.      COPY Rep Basic (i.e. recurrence pattern) from the Recurring Master into Synthetic Master
563.      COPY Exclusion List from the database Recurring Master into Synthetic Master and MERGE
          with New_Exclusion_List
564.      COMPUTE all Hash values for Synthetic Master
565.      CREATE new FIG between Synthetic Master the CIGmates of the H-FIG records
566.      CREATE CIG among the three Recurring Masters

{Fan Out Creep}

567.      Fan out Recurring Master with Previous_Date_Range
568.      Fan out Recurring Master with Current_Date_Range
569.      IF two date arrays are NOT identical, THEN MARK CIG with Fan_Out_Creep flag
570.      MARK all Records in H_File Recurring Master FIG and Synthetic Master FIG as
          Dependent_FIG
571.  END LOOP
```

FIG. 13

Pseudo Code for EXPANDING ID_BASED CIGs

```
600.  For each H_record,
601.      IF single record CIG, THEN GO TO END LOOP
602.      IF triple record CIG, THEN REMOVE CIG records from their SKGs
603.      IF Dependent_FIG, THEN GO TO END LOOP
604.      IF record needed to make triple has to be from a DB with unique ID, THEN GO TO END
          LOOP
605.      For all members of SKG to which H_record belongs
606.          IF Non_Key_Field_Hash of H_record and SKG_record Match, THEN
607.              IF Exact Match of all fields with H item THEN Strong_Match is found END
                  IF
608.          ELSE
609.              IF H_Record is a Recurring Master, THEN Find Fanned Instance (Table
                  Recurring Master/Instance Match) which is Strong_Match
610.          END IF
611.      END LOOP
612.      IF Strong_Match is found AND IF the SKG_Record is Weak_Match member of a CIG, THEN
613.          REMOVE SKG Record from Weak_Match CIG AND Seek Alternate Weak_Match for
              the CIG
614.          ADD SKG record to Current doubleton CIG AND Record for the Weak_Match_CIG
615.          REMOVE all records in CIG from SKG
616.      END IF
617.      IF Strong Match is NOT found, THEN FIND Weak_Match
618.      IF Weak Match is found, THEN create Weak_CIG
619.          ELSE REMOVE all records in CIG from SKG
620.      END IF
621.  END LOOP
```

FIG. 14

Pseudo Code for Finding Weak Matches for a Record

```
622.   FOR EVERY Record in SKG
623.       IF (SKG record is from same database as records for which match is sought OR
624.           SKG record already is a Weak_Match record in a CIG OR
625.           SKG record is a Dependent_FIG OR
626.           SKG record is Non_Recurring AND records for which is sought are not, OR
627.           SKG record is Recurring AND records for which is sought are not)
628.       THEN
629.           GO TO END LOOP
630.       ELSE
631.           If recurring item OR Key_Date_Field match Exactly, THEN Weak_Match is found
632.       END IF
633.   END LOOP
```

FIG. 15

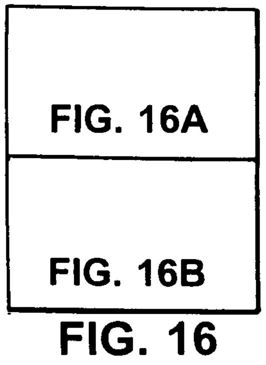


FIG. 16

Pseudo Code for Finding Matches between Recurring items and Non_Unique ID Bearing Instances

```
650.  IF Instances' database does not have unique ID OR synchronizing from scratch THEN CONTINUE
651.          ELSE EXIT
652.  END IF
653.  FOR any Recurring_Master not in Instances database,
654.          Fan out Recurring_Master for Previous_Date_Range into Previous_Date_Array
655.          MARK all entry as Previous_Date_Range_Instance
656.          Fan out Current_Recurring_Master for Current Data Range into Current_Dates_Array
657.          MARK all entries as Current_Date_Range_Instance
658.          MARK records in Exclusion_List as EXCLUDED_Dates
659.          MERGE Exclusion_List, Previous_Date_Array and Current_Date_Array into
              Merged_Date_Array
660.          CREATE Slave_Date_Array
661.          FOR EVERY item in SKG of Recurring_Master
662.                  IF Recurring item OR NOT Instances database record, THEN GO TO END LOOP
663.                  IF Start_Date of SKG record Matches an Entry in Merged_Date_Array THEN STORE
                      in Slave_Array WORKSPACE record number of SKG record AND
                      Merged_Date_Array in Slave Array
664.          END LOOP
665.          FOR EVERY Unique Non_Date Hash of Slave_Array records
666.                  FIND Slave_Array records with matching Non_Date Hash
667.                  COUNT number of matches
668.          END LOOP
669.          FIND the largest number of match counts
670.          IF largest is less than 30% of number of unexcluded instances of Master Recurring, THEN
              EXIT
```

FIG. 16A

```
671.            IF Match equals one, THEN IF NOT exact match, THEN EXIT
672.            CREATE Homogenous_Instance_Group from the records which have the same Non_Date_Hash
                value as the largest match
673.            CREATE new record Synthetic_Master in WORKSPACE
674.            COPY Basic Repeat Pattern of Recurring_Master into Synthetic Master
675.            COPY Other values from 1st item of Homogeneous Instance Group into Synthetic Master
676.            CREATE Synthetic_Master Exclusion_List based on differences between Merged_Date_Array
                and Homogeneous_Instance_Group
677.            COMPUTE Hash values for Synthetic_Master
678.            ADD Synthetic_Master to CIG of Recurring_Master
679.            CREATE Synthetic_Master FIG from all Homogeneous_Instances_Group item
680.            FOR EVERY Homogeneous_Instances_Group_item,
681.                IF Weak_match in another CIG, THEN REMOVE from CIG AND FIND New WEAK
                    MATCH for that CIG
682.                REMOVE from its SKG
683.                MARK as Dependant_FIG
684.            END LOOP
685.            IF dates in Previous_Date_Array which are not in Current_Date_Array OR Vice_versa THEN
                MARK CIG Fan_Out_Creep Flag (for unload time)
686.    END LOOP
```

FIG. 16B

Pseudocode for Completing SKG Analysis

```
700.  IF A_database AND B_database are unique ID bearing DBs, THEN REMOVE ALL remaining H_items
      from SKGs
702.  END IF
703.  FOR ALL SKGs in WORKSPACE
704.          IF SKG is singleton, THEN GO TO END LOOP
705.          FOR ALL items in Current_SKG
706.                  IF item is Weak_Match AND part of ID_based pair, THEN REMOVE from SKG
707.          END LOOP
708.          FOR ALL records in Current_SKG begining with H_Records
709.                  Call Set CIG_Max_Size in F igure 18
710.                  FIND Strong Match or Master/Instance Match between Non_ID bearing database
                      record and H_Records
711.                  IF FOUND, THEN ADD to CIG
712.                          ELSE IF FIND Strong_Match in SKG between BA and B database records
                              THEN Attach records together as CIG END IF
713.                  END IF
714.                  IF CIG_Size = CIG_MAX_Size, THEN REMOVE ALL CIG members from SKG
715.          END LOOP
716.          IF CIG_Max_Size = 3, THEN
717.                  FOR EVERY two record CIG in SKG,
718.                          FIND Weak_Match (Same Key_Date_Field and Same Recurrence Level)
719.                          IF Weak_Match item from opposing DB, THEN ADD to CIG
720.                          REMOVE records in CIG from SKG
721.                  END LOOP
722.          END IF
723.          FOR EVERY SKG item
724.                  FIND Weak_Match (Same Key_Date_Field and Same Recurrence Level)
725.                  IF FOUND, THEN ADD to CIG and REMOVE from SKG
726.          END LOOP
727.  END LOOP
```

FIG. 17

Pseudocode for setting Maximum CIG Size for Every CIG analyzed in Fig. 17.

```
750.    CIG_Max_Size = the number of non-unique ID bearing applications +1
751.    If the CIG_Max_size = 1 and CIG is not a H_Record THEN CIG_MAX_Size = 2
```

FIG. 18

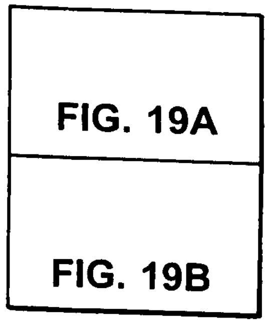


FIG. 19

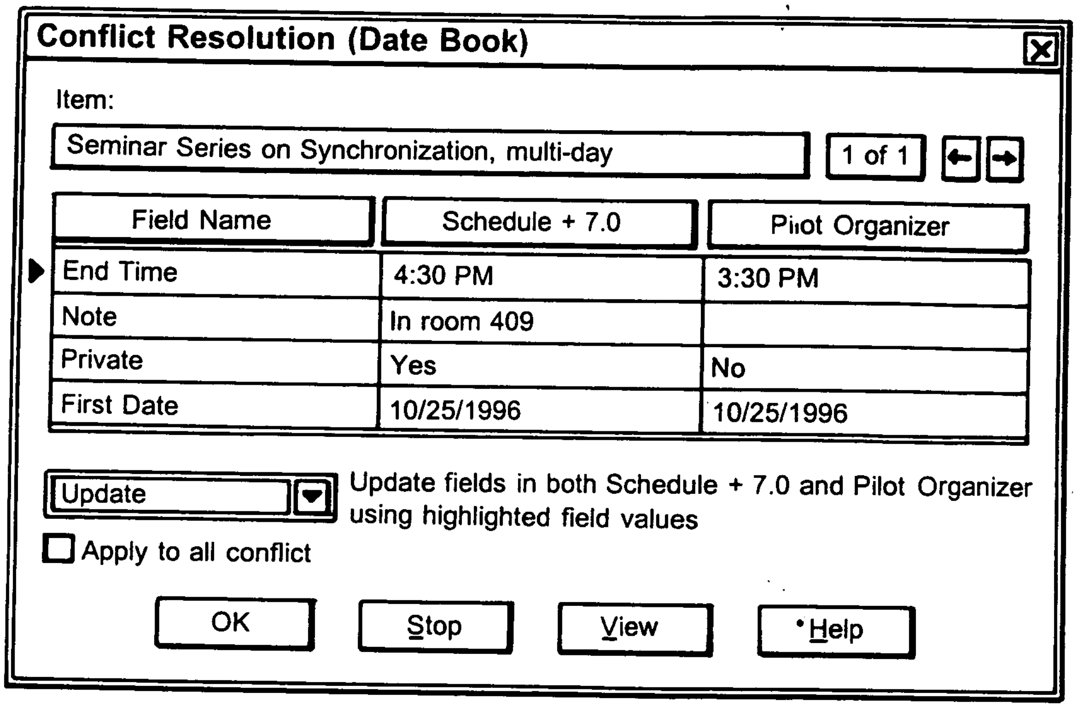
Pseudo Code for setting CIG types

```
800.    FOR EVERY CIG
801.        IF CIG Size is 1, THEN
802.            DETERMINE origin of the CIG record
803.            IF H_Record, THEN CIG_Type = 010
804.            IF B_Record, THEN CIG_Type = 001
805.            IF A_Record, THEN CIG_Type = 100
806.        END IF
807.        IF CIG Size is 2, THEN
808.            COMPARE the two CIG records
809.            IF two members are the same, THEN
810.                DETERMINE the origin of the CIG records
811.                IF B_Record and H_Record, THEN CIG_Type = 011
812.                IF A_Record and H_Record, THEN CIG_type = 110
813.                IF B_Record and A_Record, THEN CIG_type = 101
814.            END IF
815.            IF two records are different, THEN
816.                DETERMINE the origin of the CIG records
817.                IF B_Record and H_Record, THEN CIG_Type = 012
818.                IF A_Record and H_Record, THEN CIG_type = 210
819.                IF B_Record and A_Record, THEN CIG_type = 102
820.            END IF
```

FIG. 19A

```
821.        END IF
822.        IF CIG_Size = 3, THEN
823.            COMPARE records
824.            DETERMINE origins of records
825.            IF ALL records are the same, THEN CIG_Type = 111
826.            IF A_Record different from the other two and B_Record= H_Record,     THEN
                CIG_Type = 211
827.            IF B_Record different from the other two and A_Record= H_Record,     THEN
                CIG_Type = 112
828.            IF H_Record different from the other two and B_Record= A_Record,     THEN
                CIG_Type = 212
829.            IF ALL records are different, THEN CIG_Type = 213
830.        END IF
831.    END LOOP
```

FIG. 19B


Conflict Resolution (Date Book)

Item: Seminar Series on Synchronization, multi-day — 1 of 1

Field Name	Schedule + 7.0	Pilot Organizer
End Time	4:30 PM	3:30 PM
Note	In room 409	
Private	Yes	No
First Date	10/25/1996	10/25/1996

Update — Update fields in both Schedule + 7.0 and Pilot Organizer using highlighted field values

☐ Apply to all conflict

FIG. 20

Pseudocode for Merging Exclusion Lists

```
850.    FOR ALL Recurring Masters,
851.            IF CIG_Type is 102 and conflict is unresolved THEN GO TO END LOOP
{Changing CIG TYPE}
852.            COMPARE Exclusion_Lists of Current_CIG A and B records to determine Exclusion instances
                which appear in only one of the two records (i.e. One_Side_Only_Exclusion)
853.            IF None THEN do nothing
854.                    ELSE IF One_side_only_Exclusion in A_Record but not in B THEN USE Table in
                        FIG. 22 to Convert CIG_Type
855.                    ELSE IF One_Side_Only_Exclusion in B record but not in A THEN USE Table in
                        FIG. 23 to Convert CIG_Type
856.                    ELSE IF One_Side_Only_Exclusion in both records, THEN USE Table in FIG. 24 to
                        convert CIG_Type
857.            END IF
858.    END LOOP
```

FIG. 21

Old CIG + choice	new CIG	new Conflict Resolution Choice	Other Instructions & Comments
101	102	ADB Wins	
111	211		
112	132		Replace H_Record with a copy of the B_Record, plus the ADB Exclusion List
211	211		
212	213	ADB Wins	
132	132		Copy ADB ExclusionList into P-Item
102-Ig	102	Ignore	
102-SW	102	ADB Wins	
102-TW	132		Create H_Record by copying the B_Record, plus the ADB Exclusion List
213-Ig	213	ADB Wins, Excl Only	The Excl Only flag is set so that only the Exclusion List will be updated. Other BDB Fields will remain unchanged.
213-SW	213	ADB Wins	
213-TW	132		Replace P-Item with a copy of the B_Record, plus the ADB Exclusion List

(Ig for Ignore, SW for ADB Wins, or TW for BDB Wins).

FIG. 22

Old CIG + choice	new CIG	new Conflict Resolution Choice	Other Instructions & Comments
101	102	BDB Wins	
111	112		
112	112		
211	132		Replace P-Item with a copy of the A_Record, plus the BDB Exclusion List
212	213	BDB Wins	
132	132		Copy BDB ExclusionList into P-Item
102-Ig	102	Ignore	
102-SW	132		Create P-Item by copying A_Record, plus the BDB Exclusion List
102-TW	102	BDB Wins	
213-Ig	213	BDB Wins, Excl Only	The Excl Only flag is set so that only the Exclusion List will be updated. Other ADB Fields will remain unchanged.
213-SW	132		Replace P-Item with a copy of the A_Record, plus the BDB Exclusion List
213-TW	213	BDB Wins	

(Ig for Ignore, SW for ADB Wins, or TW for BDB Wins)

FIG. 23

Old CIG + choice	new CIG	new Conflict Resolution Choice	Other Instructions & Comments
101	132		Create P-Item by copying B_Record, plus the Merged Exclusion List
111	132		Copy Merged Exclusion List into P-Item.
112	132		Replace P-Item with a copy of the B_Record, plus the Merged Exclusion List
211	132		Replace P-Item with a copy of the A_Record, plus the Merged Exclusion List
212	132		Replace P-Item with a copy of the B_Record, plus the Merged Exclusion List
132	132		Copy Merged ExclusionList into P-Item
102-Ig	102	Ignore	
102-SW	132		Create P-Item by copying A_Record, plus the Merged Exclusion List
102-TW	132		Create P-Item by copying B_Record, plus the Merged Exclusion List
213-Ig	132	Excl Only	Copy Merged ExclusionList into P-Item. The Excl Only flag is set so that only the Exclusion List will be updated. Other ADB and BDB Fields will remain unchanged.
213-SW	132		Replace P-Item with a copy of the A_Record, plus the Merged Exclusion List
213-TW	132		Replace P-Item with a copy of the B_Record, plus the Merged Exclusion List

(Ig for Ignore, SW for ADB Wins, or TW for BDB Wins)

FIG. 24

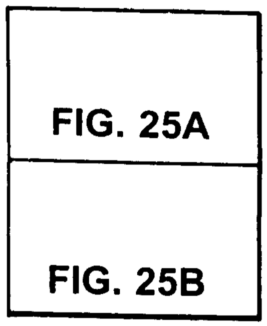


FIG. 25

Pseudo Code for Unloading Records from WORKSPACE to a database for non_rebuild_all database

```
899.    FOR all Recurring Masters which require Fanning and Outcome is UPDATE or DELETE, call
        Synchronizer Function Fanning for Unloading, Fig.27
900.    COUNT RECORDS to be Unloaded by examining all CIGs
901.     FOR EVERY RECORD to be Unloaded
{DETERMINE OUTCOME}
902.            IF MARKED GARBAGE, THEN SKIP
903.            IF BYSTANDER AND NOT History File Unload, THEN SKIP
904.            IF WRONG_SUBTYPE AND NOT Rebuild_All Translator, THEN SKIP
905.            IF Recurring_Master THEN IF Fanned for the database THEN UNLOAD Instances when
                unloading END IF
906.                    ELSE UNLOAD Recurring Master when unloading
907.            END IF
908.            LOOK UP Outcome_Sync (i.e., Unload Instructions) in Fig. 26 Table based on CIG_TYPE]
909.            IF Date Range Limited Database and Date_Range_Option = LENIENT, THEN
910.                    IF RECORD is Out of Current_Date_Range AND Outcome is not DELETE, THEN
                        SKIP Record
911.            ELSE IF Date Range Limited Database and Date_Range_Option = STERN, THEN
912.                    IF RECORD is Out of Current_Date_Range, THEN Outcome=DELETE
913.            END IF
914.            IF Outcome = DELETE, THEN
915.                    Get Info Required for this database to DELETE RECORD
916.                    (may include unique ID, Record ID, or the original values of one or more key fields, to
                        look up record so that it can be deleted)
917.                    DELETE Record
918.                    SEND Synchronizer SUCCESS/FAILURE FLAG
919.            END IF
```

FIG. 25A

```
920.            IF Outcome = ADD, THEN
921.                    GET Current values of all Fields, from Synchronizer
                        (Synchronizer maps for A database based on B→A, in response to each request)
922.                    CREATE new RECORD in DB
923.                    IF Unique_ID DB, THEN GET Unique_ID
924.                    SEND to Synchronizer (Success FLAG with any Unique_ID) OR (Failure Flag)
925.                    Synchronizer: Store Unique_ID in WORKSPACE
926.            END IF
927.            IF Outcome is UPDATE THEN GET  Current values to be unloaded and original values loaded
                from database from Synchronizer
928.                    COMPARE and DETERMINE which Field to be updated
929.                    UPDATE fields in the record to be updated
930.                    SEND to Synchronizer (Success flag AND Unique_ID) OR (Failure Flag)
931.                    Synchronizer:  STORE Unique_ID in WORKSPACE
932.            END IF
933.    END LOOP
```

FIG. 25B

```
// Original  Current
//  Item      Item     Outcome
// -------   -------   -------
{

//--- TIFCIG_001 - 1  (0) // item is present in BDB only

   B,          B,        oLEAVE_ALONE,  // unloading to BDB
   B,          B,        oADD,          // unloading to ADB
  B,        B,       oSAVE,         // unloading to History File

//--- CIG_100 - 1  (1) // item is present in ADB only

  A_        A_       oADD,          // unloading to BDB
  A_        A_       oLEAVE_ALONE,  // unloading to ADB
  A_        A_       oSAVE,         // unloading to History File

//--- CIG_101 - 1  (2) // item is identical in ADB and BDB

  B_        B_       oLEAVE_ALONE,  // unloading to BDB
  A_        A_       oLEAVE_ALONE,  // unloading to ADB
  A_        B_       oSAVE,         // unloading to History File

//--- CIG_102 - 1  (3) // NEW ADB ITEM < > NEW BDB ITEM
                       // (the BDB WINS outcome is shown here)

  B_        B_       oLEAVE_ALONE,  // unloading to BDB
  A_        B_       oUPDATE,       // unloading to ADB
  A_        B_       oSAVE,         // unloading to History File

//--- CIG_111 - 1  (4) // item is unchanged across the board

  B_        B_       oLEAVE_ALONE,  // unloading to BDB
  A_        A_       oLEAVE_ALONE,  // unloading to ADB
  H_        H_       oSAVE,         // unloading to History File

//--- CIG_112 - 1  (5) // item CHANGED in BDB since last sync

  B_        B_       oLEAVE_ALONE,  // unloading to BDB
  A_        B_       oUPDATE,       // unloading to ADB
  H_        B_       oSAVE,         // unloading to History File

//--- CIG_110 - 1  (6) // item DELETED from BDB since last sync

  H_        H_       oLEAVE_DELETED, // unloading to BDB
  A_        A_       oDELETE,        // unloading to ADB
  H_        H_       oDISCARD,       // unloading to History File
```

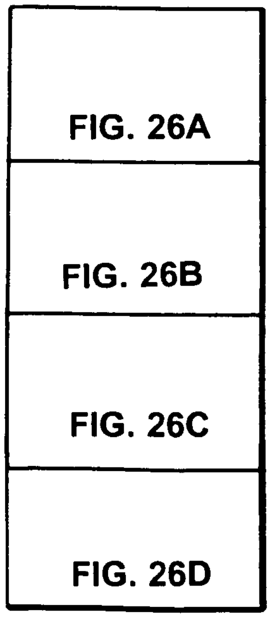


FIG. 26

FIG. 26A

```
//--- CIG_211 - 1  (7) // item CHANGED in ADB since last sync

  B_      A_     oUPDATE,       // unloading to BDB

  A_      A_     oLEAVE_ALONE,  // unloading to ADB
  H_      A_     oSAVE,         // unloading to History File

//--- CIG_212 - 1  (8) // item CHANGED IDENTICALLY in Src & BDB

  B_      B_     oLEAVE_ALONE,  // unloading to BDB
  A_      A_     oLEAVE_ALONE,  // unloading to ADB
  H_      A_     oSAVE,         // unloading to History File

//--- CIG_213 - 1  (9) // item CHANGED DIFFERENTLY in Src & BDB
                       // (the BDB WINS outcome is shown here)

  B_      B_     oLEAVE_ALONE,  // unloading to BDB
  A_      B_     oUPDATE,       // unloading to ADB
  H_      B_     oSAVE,         // unloading to History File

//--- CIG_210 - 1  (10) // CHANGED in ADB, DELETED from BDB

  A_      A_     oADD,          // unloading to BDB
  A_      A_     oLEAVE_ALONE,  // unloading to ADB
  H_      A_     oSAVE,         // unloading to History File

//--- CIG_011 - 1  (11) // item DELETED from ADB since last sync

  B_      B_     oDELETE,       // unloading to BDB
  H_      H_     oLEAVE_DELETED, // unloading to ADB
  H_      H_     oDISCARD,      // unloading to History File

//--- CIG_012 - 1  (12) // DELETED from ADB, CHANGED in BDB

  B_      B_     oLEAVE_ALONE,  // unloading to BDB
  B_      B_     oADD,          // unloading to ADB
  H_      B_     oSAVE,         // unloading to History File

//--- CIG_010 - 1  (13) // item DELETED from both ADB & BDB

  H_      H_     oLEAVE_DELETED, // unloading to BDB
  H_      H_     oLEAVE_DELETED, // unloading to ADB
  H_      H_     oDISCARD,      // unloading to History File

//--- CIG_132 - 1  (14) // 102 conflict resolved interactively
                        // to a "compromise" value stored in P-item
                        // outcome is always UPDATE BOTH

  B_      H_     oUPDATE,       // unloading to BDB
  A_      H_     oUPDATE,       // unloading to ADB
  A_      H_     oSAVE,         // unloading to History File
```

FIG. 26B

```
  // Original  Current
  //  Item      Item     Outcome
  // -------   -------   -------
  {

//-------------------- Entries for _OUTCOME_SYNC_BDB_WINS

  //--- _CIG_TYPE_102   // NEW ADB ITEM < > NEW BDB ITEM

     B_        B_       oLEAVE_ALONE,  // unloading to BDB
     A_        B_       oUPDATE,       // unloading to ADB
     A_        B_       oSAVE,         // unloading to History File

  //--- _CIG_TYPE_213   // item CHANGED DIFFERENTLY in Src & BDB

     B_        B_       oLEAVE_ALONE,  // unloading to BDB
     A_        B_       oUPDATE,       // unloading to ADB
     H_        B_       oSAVE,         // unloading to History File

//-------------------- Entries for _OUTCOME_SYNC_ADB_WINS

  //--- _CIG_TYPE_102   // NEW ADB ITEM < > NEW BDB ITEM

     B_        A_       oUPDATE,       // unloading to BDB
     A_        A_       oLEAVE_ALONE,  // unloading to ADB
     B_        A_       oSAVE,         // unloading to History File

  //--- _CIG_TYPE_213   // item CHANGED DIFFERENTLY in Src & BDB

     B_        A_       oUPDATE,       // unloading to BDB
     A_        A_       oLEAVE_ALONE,  // unloading to ADB
     H_        A_       oSAVE,         // unloading to History File

//-------------------- Entries for IGNORE (LEAVE UNRESOLVED)

  //--- _CIG_TYPE_102   // NEW ADB ITEM < > NEW BDB ITEM

     B_        B_       oLEAVE_ALONE,  // unloading to BDB
     A_        A_       oLEAVE_ALONE,  // unloading to ADB
     B_        B_       oDISCARD,      // unloading to History File

  //--- _CIG_TYPE_213   // item CHANGED DIFFERENTLY in Src & BDB

     B_        B_       oLEAVE_ALONE,  // unloading to BDB
     A_        A_       oLEAVE_ALONE,  // unloading to ADB
     H_        H_       oSAVE          // unloading to History File

}; //---- TableAfterILCR
```

FIG. 26D

```
//--- CIG_13F - 1  (15) // 132 UPDATE-BOTH
                       // which has been Fanned To BDB

    B_      B_      oDELETE,      // unloading to BDB
    A_      H_      oUPDATE,      // unloading to ADB
    A_      H_      oSAVE         // unloading to History File

   // Note that we delete the recurring master on the BDB Side;
   // fanned instances take its place.

};
```

The table entries above for CIG_102 and CIG_213 are only relevant when the Conflict Resolution Option is set to BDB WINS. If the Conflict Resolution Option is set to IGNORE or ADB WINS then those table entries are adjusted accordingly. For IGNORE we use the following table entries:

```
// Original  Current
//  Item      Item     Outcome
// -------   -------   -------
//--- _CIG_TYPE_102   // NEW ADB ITEM < > NEW BDB ITEM

    B_      B_      oLEAVE_ALONE,  // unloading to BDB
    A_      A_      oLEAVE_ALONE,  // unloading to ADB
    B_      B_      oDISCARD,      // unloading to History File

//--- _CIG_TYPE_213   // item CHANGED DIFFERENTLY in Src & BDB

    B_      B_      oLEAVE_ALONE,  // unloading to BDB
    A_      A_      oLEAVE_ALONE,  // unloading to ADB
    H_      H_      oSAVE,         // unloading to History File
```

And for ADB WINS we use the following table entries:

```
// Original  Current
//  Item      Item     Outcome
// -------   -------   -------

//--- _CIG_TYPE_102   // NEW ADB ITEM < > NEW BDB ITEM

    B_      A_      oUPDATE,       // unloading to BDB
    A_      A_      oLEAVE_ALONE,  // unloading to ADB
    B_      A_      oSAVE,         // unloading to History File

//--- _CIG_TYPE_213   // item CHANGED DIFFERENTLY in Src & BDB

    B_      A_      oUPDATE,       // unloading to BDB
    A_      A_      oLEAVE_ALONE,  // unloading to ADB
    H_      A_      oSAVE,         // unloading to History File
```

When the NOY option is in effect, CIG-specific conflict outcomes are recorded in the CIG members' flag bits. When this is the case the following lookup table is used:

```
static unsigned char TableAfterILCR [_SYNC_OUTCOME_COUNT]
                                    [AFTER_ILCR_CIG_TYPE_COUNT]
                                    [SYNC_UNLOAD_PHASE_COUNT]
                                    [3] =
```

FIG. 26C

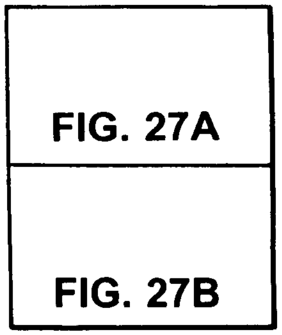


FIG 27

```
FANNING Recurring_Items for Unloading (for A DB)

          Fan Pattern for paper Date Range (Fig. XX)

950.      IF Outcome is UPDATE, THEN
951.          IF (CIG A_Record was a Recurring Master but now to be fanned and CIG B_Record is a
              Recurring Master) THEN IF CIG_Type = 132 THEN CIG_Type = 13F
952.              GOTO Fanning For ADD
953.              ELSE
954.                  SET A_Record CIG_Type to 100
955.                  SET B_Record CIG_Type to 001
956.                  SET H_Record CIG_Type to 010
957.                  MARK A_Record with DELETE_ME Flag
958.                  GOTO Fanning for Add
959.              END IF
960.          END IF
961.          IF (CIG A_Records were fanned previously and Fanned now) AND (CIG B_record recurring),
              THEN
962.              FOR ALL A items in Synthetic Master FIG
963.                  STORE Start_Date in Date_Array_Temporary
964.              END LOOP
965.              Fan_Out_Recurring_Pattern of B Master
966.              COMPARE Date_Array_Temp with Fan_Out_Date_Array
967.              MARK Dates which NOT IN Fan_Out_Date Array with DELETE_Me Flag
968.              IF Date NOT IN Date_Array_Temp, THEN
969.                  CREATE WORK SPACE Record by Copy Recurring_Master but Omit Rep
                      Basic, Rep Excl, Unique ID Field
970.                  SET Start_Date, End_Date, Alarm_Date to values for Current Instance
971.                  Compute Hash
972.                  MARK Fanned_For_A
973.              END IF
```

FIG. 27A

```
974.            IF Date in Date_Array_Temp AND Fan_Out_Date_Array THEN
975.                Compare Non_Date Hash to Synthetic Master Non_Date_Hash
976.                IF Same, THEN MARK Leave_Alone
977.                ELSE MARK UPDATE END IF
978.            END IF
979.        END IF
980.        IF (A_Record Recurring previously and to be Fanned now) AND (CIG B_Record is Instances)
            THEN
981.            MARK CIG items as Garbage
982.            MARK FIG items of CIG H_record as Garbage
983.            MAKE FIG items of CIG B_record singletons
984.        END IF
985.    ELSE [Fanning For Add]
986.        Fan out Recurrence Pattern
987.        For each Date in Fan_Out_Date_Array
988.            COPY Master item into new WORKSPACE Record except Omit Rep_Basic,
                Rep_Exclusion, and Unique ID
989.            Use Date for Start Date and End Date
990.            Set Alarm Date, if necessary
991.            Compute Hash Values
992.            Attach to Recurring_Master FIG
993.            Set Fanned_for_A Flag
994.        END LOOP
995.    END IF
```

FIG. 27B

Pseudocode for Unloading History FILE

```
1000.  ERASE previous History File and CREATE new one
1001.  FOR EVERY CIG in WORKSPACE
1002.          Look up in Fig. 26 Table based on CIG_Type AND DETERMINE whether should be unloaded
               into the History File
1003.          IF NO THEN GOTO END LOOP
1004.          IF Exclusion_List_Only Flag is set when merging of Exclusion_List THEN REPLACE History
               RECORD Exclusion_List with new Merged Exclusion_List
1005.          Clear all Flag bits except for Recurring_Record flag
1006.          SET origin flag to History_Record
1007.          Clear FIG, SKG and CIG words
1008.          STORE Applicable Unique IDs
1009.          IF Recurring item, THEN STORE ALL ID_Bearing FIG records AND SET their FIG in
               History_File to keep them together
1010.          STORE Record in History File
1011.          IF current record is a recurring master for an ID-bearing FIG THEN STORE FIG Records(i.e.
               all Fanned Instances) in the History File, with the FIG linkage words set in the History File to
               hold the FIG together.
1012.  END LOOP
1013.  STORE Field Lists, Application Names, Database Names, Current Date Range,
```

FIG. 28

	How Item is stored in Other Database	How stored in Unloader's Database Before Fanning For Update	How stored in Unloader's Database After Fanning For Update
1	Master	Master	Instances
2	Master	Instances	Instances
3	Instances	Master	Instances

FIG. 29

```
1050.   Verify History File
1051.           If verified, Then Proceed as Fast Synch
1052.           If not, Then Proceed as Synchronization from Scratch load all record in database

1053.   If Fast Synch
1054.           LOAD records into the Workspace.  Map if necessary
1055.           Sanitize Records not marked as Deletion
1056.           Orientation analysis (Fig. 11).
1057.           For each H_Record, analyze the CIG that the H_Record belongs to.
1058.                   IF the H_Record's CIG contains no Record from the Fast Synchronization database,
                        THEN CLONE the H-Item, label it a Fast Synchronization Record, and add it to the
                        H_Record's CIG.
1059.                   If the H_Record's CIG contains a Fast Synchronization record that is marked as a
                        Deletion, it is now removed from the CIG.
1060.                   If the H_Record's CIG contains a non-Delete Fast Synchronization Record, then do
                        nothing.
1061.   .       END LOOP
```

FIG. 30

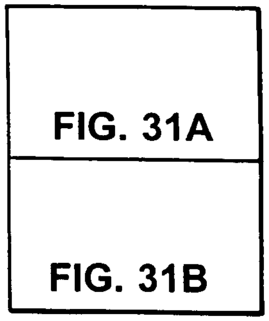


FIG. 31

```
1150.  Verify History File
1151.          If verified, Then Proceed as Fast Synch
1152.          If not, Then Proceed as Synchronization from Scratch

1153.  IF synchronization from scratch
1154.          IF record outside of current_date_range THEN MARK record as out-of-range

1155.  If Fast Synch
1156.          Load History File into Workspace
1157.          MARK History File records outside of previous_date_range as Bystander
1158.          Load All Fast Synchronization Records into the Workspace; mapped if necessary.
1159.          SANITIZE Records which are not DELETEs
1160.          Orientation analysis  (Fig. 11).
1161.          If Added Fast Synchronization record is out of current date range THEN MARK Out-Of_Range
1162.          If Changed or deleted Fast Synchronization record in a CIG with Bystander H_Record, MARK
               the Bystander record as Garbage
```

FIG. 31A

```
1163.   For each H_Record, analyze the CIG that the H_Record belongs to.
1164.       If the H_Record's CIG contains no Record from the Fast Synchronization database,
            then make a clone of the H-Item, label it a Fast Synchronization Record, and adding it
            to the H_Record's CIG.
1165.       If H_Record is not a Bystander, THEN Make a clone of H_Record, mark as Fast
            Synchronization record, and Add to CIG
1166.       IF H_Record is Bystander THEN
1167.           IF outside of Current date range THEN Do Nothing
1168.           ELSE {Within Current Date Range}
1169.               Mark H_Record as Garbage, Clone H_Record and Mark it as from
                    Fast Synchronization database
1170.           END IF
1171.       END IF
1172.       If the H_Record's CIG contains a Fast Synchronization record that is marked as a
            deletion, it is now removed from the CIG.
1173.       If the H_Record's CIG contains a non-deletion Fast Synchronization Record, then do
            nothing.
1174.       Any Fast Synchronization records which are not joined to any H_Record's CIG
            represent additions and no transformation is required.
1175.   END LOOP
```

FIG. 31B